Advances in Engineering Education

WINTER 2009

Blended Teaching and Learning of Computer Programming Skills in Engineering Curricula

ABBAS EL-ZEIN
School of Civil Engineering
University of Sydney, NSW 2006, Australia
E-mail: aelzein@usyd.edu.au

TIM LANGRISH
School of Chemical and Biomolecular Engineering
University of Sydney, NSW 2006, Australia
E-mail: T.Langrish@usyd.edu.au

NIGEL BALAAM
School of Civil Engineering
University of Sydney, NSW 2006, Australia
E-mail: n.balaam@usyd.edu.au

ABSTRACT

Many engineering schools include computer programming as part of a first-year course taught to large engineering classes. This approach is effective in rationalizing resources and improving the cost-effectiveness of course delivery. In addition, it can lead to wholesale improvements in teaching and learning. However, class sizes and the variety of student backgrounds can lead to difficulties in achieving learning outcomes. Flexible learning has been shown to be potentially effective in addressing such issues. We describe the design and development of a WebCT-based self-practice online tool (SPOT) to support student learning of programming. The tool is divided into three components: a) programming syntax, b) understanding the way computer programs work and c) writing computer programs. We discuss the integration of the tool into the learning flow and its role in assessment. We present qualitative and quantitative data on student reactions to the tool and its usefulness in achieving learning outcomes cost-effectively.

INTRODUCTION

Flexible and blended modes of learning have been adopted in many university courses around the world. Flexible learning refers to curricular environments where students can access learning resources, on- or off-campus, at times and in contexts that are suited to the student rather than

the teaching staff, whereas blended learning is a study environment which combines face-to-face and online learning [e.g., 1]. A new paradigm of online education has spawned a rich literature on the effectiveness and efficiency of various forms of electronic teaching tools, from full online courses [e.g., 2] to web-assisted, lecture-based courses [e.g., 3, 4]. The ability of these modes of teaching and learning to achieve desired engineering learning outcomes and their efficiency in achieving that aim remain open questions. Evidence points to an improvement in learning efficiency, although students with access to online resources are not necessarily more likely to achieve learning outcomes [5–7]. Although the number of distance courses has risen significantly over the last decade, mixed modes of delivery, with face-to-face settings supported by online tools, remain the dominant form of online learning on campus. There is clearly a need in the literature for greater exploration of flexible modes of learning, including e-tools, when teaching computational skills to engineering students.

Programming skills are now deemed essential in most engineering schools. Both structured languages, such as FORTRAN and C, and computational tools such as MATLAB, have been used in engineering curricula. Hodge and Steele [8] surveyed engineering programs in the USA and found that FORTRAN had lost its dominance, and computational tools were increasingly employed by educators because of the trend towards integrating various computational functions in a single environment. At the Faculty of Engineering of University of Sydney, MATLAB was adopted in an introductory computational course (ENGG1801) for first-year engineering students for two reasons: 1. its ability to integrate programming with matrix operations and graphics and 2. the relative simplicity of its programming tools, which offer the possibility of introducing students to fundamental programming concepts without requiring them to grapple with other aspects of programming, such as dimensioning and compilation. However, the development of programming skills by first-year engineering students has proved to be a complex task, especially in large 500+ student classes, and a small but significant proportion of students (20%) have failed, in the past, to perform satisfactorily.

This paper discusses the design, development and implementation of an e-learning tool into ENGG1801 and offers a student-centered model for integrating e-learning with other course resources, including face-to-face interaction. The aim of this integration is to increase the number of students who achieve the required learning outcomes and reduce the percentage of students who fail the course. While other methods for improving learning outcomes have been suggested in the literature (e.g., a crash course preceding the main course as described by Christensen et al [9]), e-learning remains more attractive because of its potential cost-effectiveness in terms of student time and financial expenditure.

CURRICULAR CONTEXT

ENGG1801 is made of two components which run in parallel: Computer-Aided Design (CAD) with SolidWorks and programming using MATLAB. The first component occupies around 40% of the course, while the second accounts for 60%. These percentages reflect the division of hours of lectures and lab sessions, as well as assessment weights. In this paper, we focus on the programming part of the course and will not discuss the CAD component. ENGG1801 is aimed at first-year civil, aeronautical, mechanical and chemical engineering students. The numbers of students enrolled in the course have increased, from 450 in 2004 to 550 in 2007.

The programming part of the course aims to develop students' skills at writing simple computer programs that can solve simple mathematical and engineering problems. By the end of the course, students are expected to be able to write sequential programs using the following families of commands: input and output, conditional structures such as "if" and "case", loop structures such as "for" and "while", modular structures such as "functions" and "subroutines" and, finally, graphic functions intrinsic to MATLAB. Although MATLAB is used in teaching, course instructors make it clear to students that the purpose of the course is not to teach MATLAB per se, but programming more generally. Skills and programming concepts used in one sequential programming language are still valid in another, with minimal adjustment, in the same way that driving skills acquired with one car brand are transmissible to another. Students are given a one hour programming lecture per week, after which they attend a computer lab session, with around 50 students in each session, where they are asked to solve a programming problem, with help from tutors.

A number of issues arose in the first two deliveries of the course in 2004 and 2005. The first issue was related to tutor-student contact. Although three tutors were allocated for each MATLAB programming session, with a ratio of 16 students per tutor, some students clearly felt they needed more tutorial support. Given budgetary constraints, it was impossible to reduce this ratio. Instead, an additional tutorial session for programming, called a clinic session, was introduced in 2006 and was run by the lecturers, rather than the tutors. Attendance was voluntary and open to all students who needed extra support. In addition, tutors were asked to provide more pro-active guidance to students at the beginning of each session.

A second issue was related to programming quizzes. Three quizzes were given during the semester. Given the large number of students, a quiz system, introduced in 2004 and followed in 2005, had students sitting their quizzes during their lab sessions, on specially designated weeks. Tutors supervised quizzes and marked responses immediately after students finished writing their answers on the computer screen. A simple marking system (0 to 3) was used. The system was effective in that

marking was done quickly and the effort was widely distributed between tutors. There were however three drawbacks. First, students were worried about inconsistency of marking between tutors, and there was no way of guaranteeing such consistency, given the large number of tutors—despite written instructions given to tutors, face-to-face meetings between tutors and instructors prior to the quizzes, and the simplicity of the marking system. Secondly, since ten different tutorial sessions per week ran to accommodate the 500 or so students, ten different versions of each quiz had to be written. Third, exam supervision was rather difficult, despite the tutors' best efforts, given the proximity to each other of student seats in the computer lab.

A third issue, perhaps the most significant one, became clear to us during the semester in 2004, and was confirmed in the final exam and during 2005. The most difficult aspect of the course was programming. The failure rate in the course stood at around 18% and the majority of students who failed did so as a result of programming. A number of measures were taken in response to this, including changes that allow a more gradual introduction of programming concepts, as well as more exercises solved in the class and the lecture notes.

The above three issues—tutoring, assessment and learning of programming concepts—are obviously related. However, for all their complexity, it is obvious that adequately-designed e-learning resources can play a major role in addressing them. This is particularly the case given the large number of students and the inevitable budgetary constraints in any curricular activity. The question asked in small, more conventional classroom environments where the teaching and learning community consists primarily of a teacher and a few dozen students is: "how best to achieve the learning outcomes of the course?" This question is best developed in a slightly different form for larger classes and more complex teaching and learning communities, which include coordinators, instructors, tutors, administration staff, as well as a few hundred students. A more pertinent question in this case is: "what is most the cost-effective way of achieving learning outcomes among the highest possible number of the students, hence reducing the number of failures in the course?" A self-practice online tool (SPOT), which addresses all three issues raised above, has been designed, developed and evaluated as a response to this question.

SELF-PRACTICE ONLINE TOOL (SPOT): RATIONALE AND ARCHITECTURE

We developed the online tool in order to achieve the following objectives:

a. to put in place better flexible learning resources for students.
b. to help students assess their own progress and provide them with a clear path for seeking additional help.

c. to better integrate lectures and lab sessions.

d. to improve the quality of assessment through quizzes.

e. to improve the cost-effectiveness of the provision of quality teaching to students.

A database of online questions (DOQ1), with around 300 multiple-choice questions, was developed. The questions were grouped under nine categories: MS Excel basics, matrix algebra, matrix MATLAB operations, and the following sets of commands in MATLAB: text, conditional ("if" and "switch"), "for" loops, "while" loops, "function" and graphics. Each category was further divided in two groups corresponding to two levels of difficulty. Each question carried five possible answers, as well as a few lines of justification for the correct answers and usually a note on each of the incorrect answer. DOQ1 questions assessed the student's understanding of the syntax and role of each set of commands. DOQ1 was later augmented with DOQ2 and DOQ3. DOQ2 consists of multiple-choice "skeletal" questions, which present students with small programs and asks them to fill in missing commands or spot errors in the programs. DOQ3 consists of programming questions which asks students to write computer programs to solve a particular problem. Hence, DOQ1, DOQ2 and DOQ3 take the students through the process of learning programming commands, understanding how computer programs work and writing computer programs. (We will refer generically to DOQ1, DOQ2 and DOQ3, by DOQ, in the remainder of the paper). DOQ was then used to generate two WebCT tools:

a. A Self-Practice Online Tool (Spot1, Spot2 and Spot3, corresponding to DOQ1, DOQ2 and DOQ3, respectively, and collectively called Spot) that could be accessed online by students enrolled in the unit of study at any time. The student could choose a particular category and test their ability, by attempting to answer the question, checking whether they had answered correctly and get specific feedback on each answer, as well as general feedback on the question.

b. A quiz tool (QT) that would be used to run 3 quizzes over the semester. Quiz 1 would be drawn from DOQ1, quiz 2 from DOQ 1 and DOQ 2, while quiz 3 consists entirely of DOQ 3 questions.

(Note that, to avoid ambiguity, Spot refers to the self-practice part of the system, while SPOT, in upper case, denotes the whole tool, including DOQ, Spot and QT). The architecture of SPOT is shown in Figure 1. Once DOQ was developed, Spot and QT were easily set up within the WebCT environment, at no extra cost. Spot and QT were assigned a specific role within a new course learning map, developed to address the problems discussed earlier (see Figure 2). The figure shows the regular learning pattern students are expected to follow. After attending a lecture introducing a new programming concept, the students read the corresponding lecture notes and lecture slides, went to the lab session to solve the corresponding problem and attempted the corresponding Spot questions. Whenever they experienced difficulties, they could speak, one-on-one, to tutors during lab sessions, post a question on the discussion board for the course and go to the clinic session.

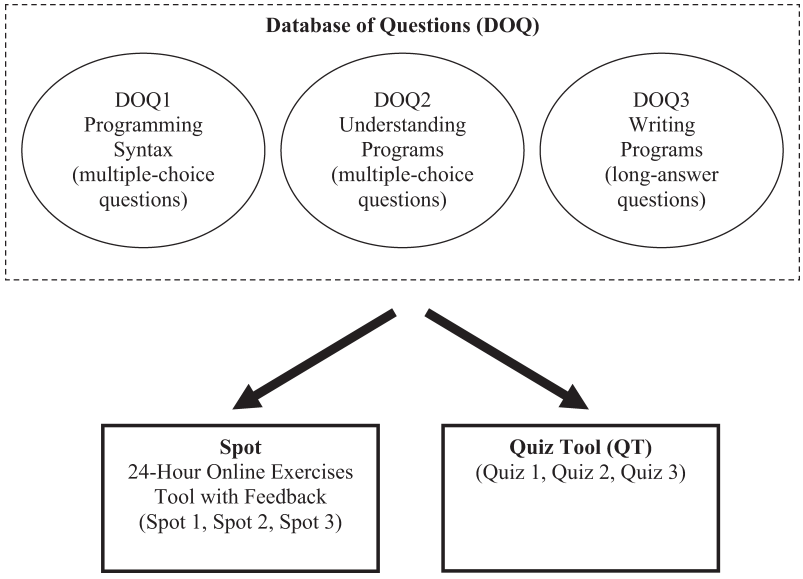


Figure 1. Architecture of SPOT.

Students could also choose to email or visit the course lecturers in their offices. Questions on the discussion board, as well as communication between tutors and instructors, helped the teaching staff keep track of the kind of difficulties arising in the class, which may then be specifically addressed by instructors during lectures. The syllabus covered first preliminary concepts such as variable type and matrix operations, followed by input and output statements. Next, if-else-end blocks, non-conditional loops and conditional loops were introduced. Finally, functions or subroutines, and graphics were covered in the course. DOQ included questions relevant to all these topics. The level of complexity of concepts clearly increased as the semester progressed and students had most difficulty with functions and related features such as local and global variables. DOQ1 and Spot1 were developed in time for semester 1 2006. DOQ2 and DOQ3, with Spot2 and Spot3, were developed in time for semester 1, 2007. The Respondus program was used for developing the questions, which were then exported into WebCT. Figure 3 shows a snapshot of the main menu of Spot, as well as a sample question from Spot1, including feedback to student's answers.

EVALUATION OF SPOT

Four forms of evaluations were used to assess the impact of SPOT, directly and indirectly: a) a student feedback survey, b) regular end-of-semester feedback scores for the course c) student

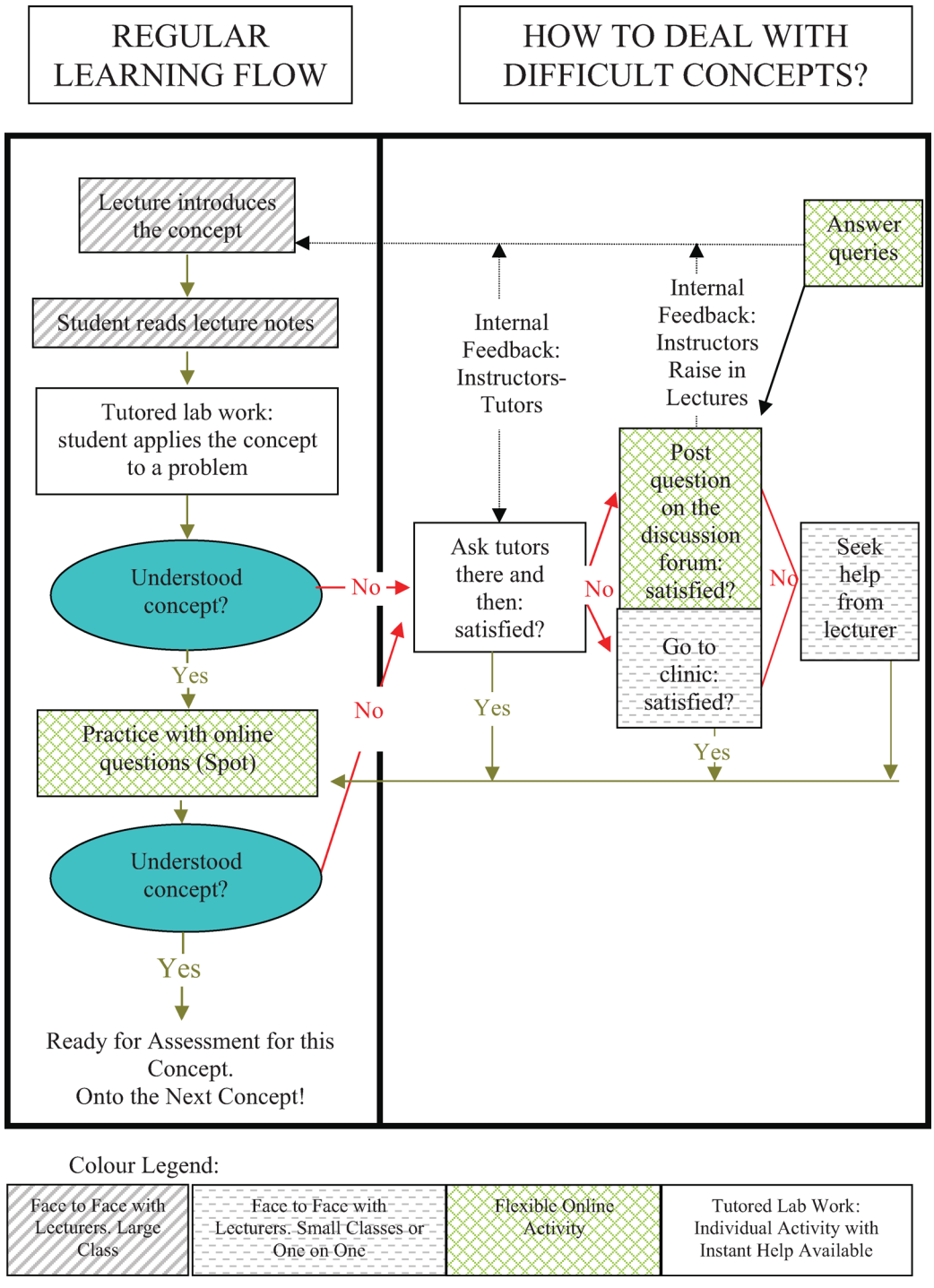
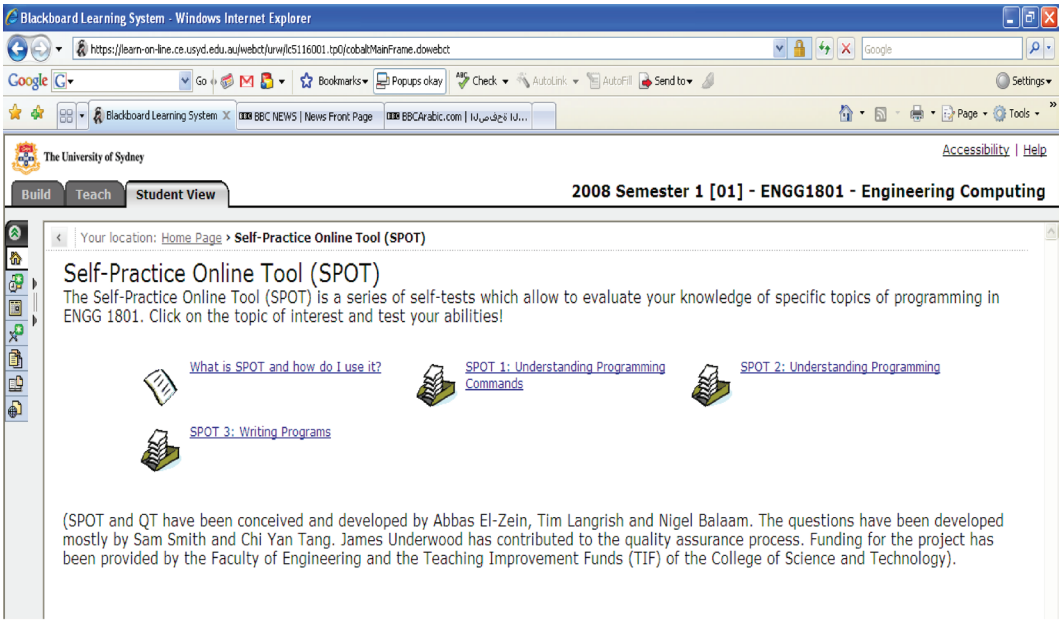
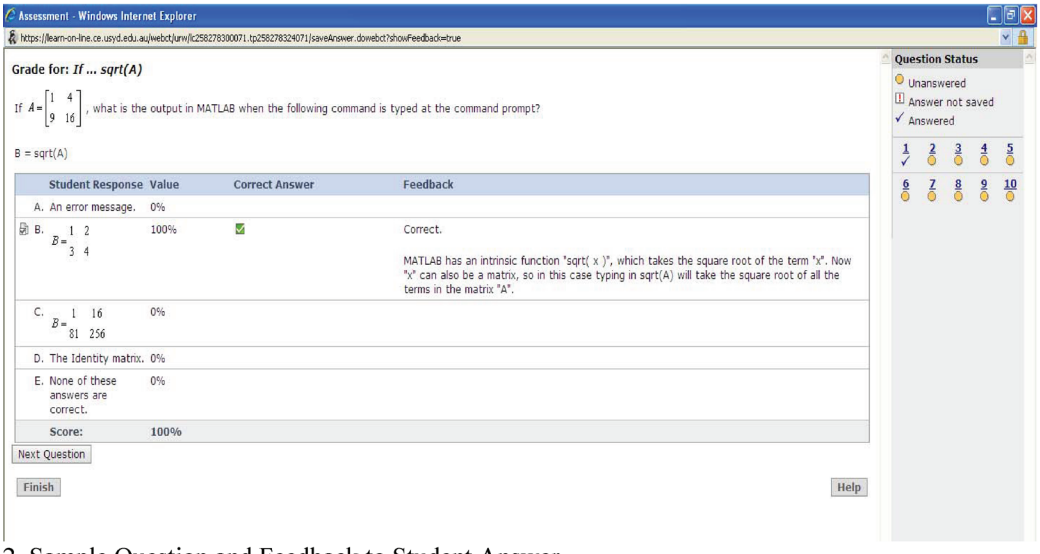


Figure 2. Learning Activity Map for the Course.



1. Main Menu of SPOT on WebCT



2. Sample Question and Feedback to Student Answer

Figure 3. Sample screen shots of SPOT main menu and sample question.

performance in quizzes and exams d) students queries and complaints about quizzes. The first survey asked specific questions about SPOT. The end-of-semester scores for the course reflected the degree of overall student satisfaction. Student performance in quizzes and exams was considered a measure of the extent to which learning outcomes were being achieved. Finally, a qualitative

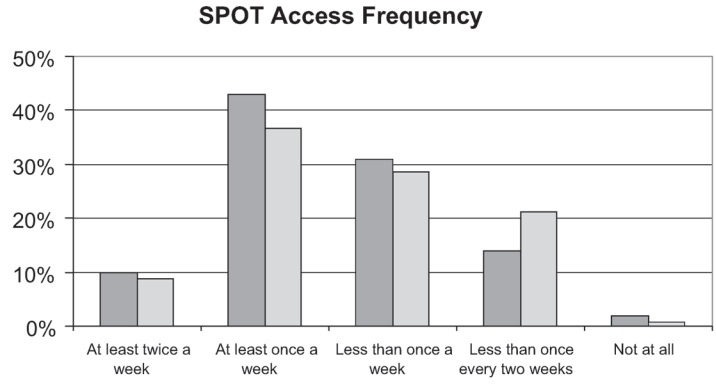
assessment of the impact of SPOT on student satisfaction with the quiz system was made by qualitatively monitoring the change in students complaints and queries about the quiz logistics and results.

Student Feedback Survey

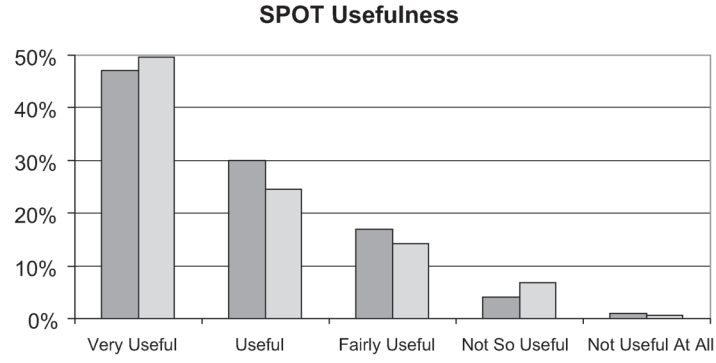
On weeks 7 in 2006 and week 9 in 2007, students were asked to fill in an anonymous questionnaire about the course, including the following three questions about SPOT (since SPOT2 and SPOT3 had not been developed by then):

1. HOW OFTEN have you accessed Spot since the beginning of the semester:
   a. At least twice a week
   b. Less than once a week
   c. Less than once every two weeks
   d. Not at all
2. HOW USEFUL did you find Spot in helping you to learn programming concepts:
   a. Very useful
   b. Fairly useful
   c. Not so useful
   d. Not useful at all
3. How useful did you find the FEEDBACK on answers in Spot?
   a. Very useful
   b. Fairly useful
   c. Not so useful
   d. Not useful at all

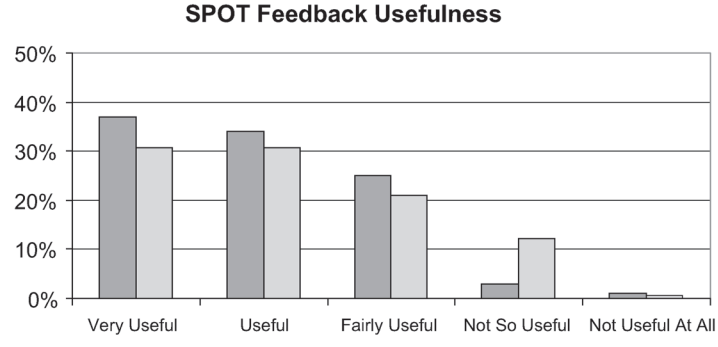
236 and 147 students responded to the questionnaire in 2006 and 2007, respectively. Total number of students was approximately 464 and 502, respectively. The difference in response rate is likely due to the higher number of opportunities given to students to answer the survey in 2006, compared to 2007. Survey statistics for the above questions are shown in Figure 4. In 2006, students had only used Spot1, while in 2007 the questionnaire referred to Spot1, Spot2 and Spot3. While 90% of respondents used Spot less than once a week in both 2006 and 2007, around 50% of respondents found Spot to be very useful and 75% found it to be useful or very useful. There was a decline in satisfaction with the feedback on Spot questions, which points to the need for more development of feedback material in Spot2 and Spot3. Nevertheless, even in 2007, more than 60% found the feedback to be useful or very useful. Clearly, possible selection bias must be kept in mind, with more involved students more likely to answer the questionnaire. This may have increased the percentage of students using Spot frequently and finding it useful. However, the consistency of the response between 2006 and 2007 increases confidence in our findings.



i. HOW OFTEN have you accessed Spot since the beginning of the semester?



ii. HOW USEFUL did you find Spot in helping you to learn programming concept?



iii. How useful did you find the FEEDBACK on answers in Spot?

Figure 4. Statistics of Student Response to SPOT (solid bar: 2006; dotted bar: 2007).

Feedback Scores for the Course

Students feedback scores for the ENGG1801 are shown in Table 1: S05, S06 and S07 for 2005, 2006 and 2007. All scores are out of a maximum 5. The number of respondents were 160, 143 and 91, in 2005, 2006 and 2007, respectively. Evaluation of the course has improved significantly

	S05	S06		S07	
Question	**Scores 2005**	**Scores 2006**	***(S06-S05)/S05***	**Scores 2007**	***(S07-S06)/S06***
The learning outcomes and expected standards of this unit of study were clear to me	2.88	3.34	*16%*	3.65	*9%*
The teaching in this unit of study helped me to learn effectively	2.42	2.82	*17%*	3.23	*15%*
This unit of study helped me develop valuable graduate attributes [e.g., 1) Research inquiry skills; 2) Communication skills; 3) Personal intellectual autonomy; 4) Ethical, social and professional understandings; 5) Information literacy]	2.91	3.04	*4%*	3.13	*3%*
The workload in this unit of study was too high	2.93	2.96	*1%*	2.6	*–12%*
The assessment in this unit of study allowed me to demonstrate what I had understood	3.22	3.41	*6%*	3.8	*11%*
I can see the relevance of this unit of study to my degree	3.74	3.81	*2%*	3.99	*5%*
It was clear to me that the staff in this unit of study were responsive to student feedback	3.18	3.58	*13%*	3.81	*6%*
My prior learning adequately prepared me to do this Unit of Study	2.42	2.8	*16%*	2.69	*–4%*
The Learning Teaching interaction helped me to learn in this Unit of study (labs and/or tutorials interaction project work)	3.34	3.56	*7%*	3.66	*3%*
My learning of this unit of study was supported by the faculty infrastructure (e.g., e-learning, labs, computers, lecture theatres, tutorial rooms)	3.84	3.61	*–6%*	4.16	*15%*
I could understand the teaching staff clearly when they explained things	3.12	3.11	*0%*	3.48	*12%*
Overall I was satisfied with the quality of this unit of study	2.9	3.26	*12%*	3.57	*10%*

(scores are out of 5; higher scores reflect more positive opinion except for question 4 about workload, where lower numbers reflect more positive opinion; "unit of study" in the questions refers to ENGG1801; t-test of significance: S05-S06: $p < 0.01$, S06-S07: $p < 0.01$, S05-S07: $p < 0.001$)

Table 1. Student Evaluation of Course in 2005, 2006 and 2007.

from 2005 to 2006 and from 2006 to 2007. Comparing the overall change in student evaluation, from 2005 to 2007, shows improvements across the entire spectrum of questions. For example, the overall score has risen from 2.9 in 2005 to 3.57 in 2007, while the score for assessment has increased from 3.22 to 3.8 over the same period. The only decline in USE score is registered for the adequacy of prior learning (high-school in this case), decreasing from 2006 to 2007 and the suitability of infrastructure from 2005 to 2006. However, the latter score improved significantly in 2007. S06 and S07 were found to be significantly higher than S05 and S06, respectively, at $p < 0.01$ using the student t-test. S07 was significantly higher than S05 at $p < 0.001$. SPOT was not the only change introduced into the course in those years and could not, therefore, be solely credited with all the improvements. However, it was certainly one of the most significant innovations.

Overall Student Performance in the Course

The percentage of students scoring less than 50% in the programming part of the course, prior to scaling, was 14%, 20% and 15% in 2005, 2006 and 2007, respectively. However, the quizzes and final examinations in 2006 and, especially 2007, have focused more on the ability to interpret computer programs and write new ones, and less on the ability to understand programming syntax. This was done in two ways: an additional quiz and a change to the final exam.

First, while students sat only two quizzes in 2005, a third quiz was introduced in 2006, and repeated in 2007, which required students to write computer programs as a solution to a given mathematical or engineering problem. Furthermore, the second quiz in 2007 focussed more strongly on evaluating the skills of students in interpreting and correcting existing computer programs. The two quizzes in 2005, on the other hand, tested the students' ability to understand programming syntax and identify errors in existing computer programs. Therefore, the quizzes in 2006 and 2007 were on the whole more difficult than in 2005.

Second, all three questions in the final exam of 2007 required students to modify existing computer programs or write new ones. By contrast, in 2005 and 2006, one of the three questions merely tested the student's ability to understand MATLAB syntax and perform basic matrix operations. As expected, the average mark was noticeably higher for this question, compared with the other two in 2005 and 2006.

It is, therefore, possible to conclude that the increase in pre-scaling rates of failure from 2005 to 2006 may be attributable to an increase in the level of difficulty of the assessment. In 2007, the level of difficulty was further increased but the failure rates have dropped from 20% to 15%. Once again, the improvement in student performance that can be inferred from these figures, is only partially attributable to SPOT. Nevertheless, at the very least, the use of flexible online resources such as

SPOT appears to be compatible with the overall drive to improve the degree to which students are achieving better programming learning outcomes.

Change in Queries about Quiz Performance

The new tools brought about a major reduction in complaints about the fairness of marking of quizzes. Even when long answers rather than multiple-choice questions were used in quiz 3, the online submissions could now be transferred to a select group of tutors who performed the marking, hence ensuring more consistency. The tool provided students with more learning resources and enhanced the assessment quality of the course. The multiple functionality of such e-tools is a key factor in their cost-effectiveness because it offsets their development costs.

Qualitative Feedback

Students were asked to comment on the course in general in an online survey, as well as the regular evaluation process conducted routinely by the university. SPOT came up frequently in students comments, almost always in a positive light. For example, a number of respondents, valued the way it helped their learning:

> "The SPOT is a great help to me with the MATLAB component. Being able to use this has aided the speed of my learning and results greatly. Still having a bit of trouble with the programming side but i guess that will come with practice eh?."

Concerning assessment, some respondents valued SPOT as a self-assessment tool:

> "The quizzes were helpful and the SPOT facility was an excellent self-assessment tool."

Others found that SPOT made assessment fairer, presumably because of the random question selection and reduction in cheating possibilities:

> "SPOT made [assessment] in this Unit of Study fair."

Finally, some respondents highlighted the role of SPOT as a source of information about quizzes:

> "The SPOT program helped me understand what is tested & how."

Criticism of the tool, when it came up, was almost exclusively concerned with the feedback provided by SPOT:

> "feedback on SPOT is generally good but for some questions it just says "incorrect"; without explaining why."

> "SPOT is a fantastic resource, however the feedback provided by SPOT is often lacking—for example not explaining why the wrong answer was wrong, or why a right answer is right. Better feed back would make SPOT both a learning tool, as well as a revision exercise."

One respondent drew an interesting connection between the effect of SPOT on assessment and its feedback content:

> "The programming component of the unit has been very well structured and devised. QT is a very well designed tool for assessing. Using the same tool for SPOT gives too much of an advantage for the quizzes, while—without feedback—not providing much real scope for further learning."

Here the insufficiency in feedback is perceived as having a negative impact on the fairness of the assessment.

Cost-Effectiveness

One of the objectives of SPOT is to provide students with a quality learning and assessment resource in a cost-effective manner. It is clear that the "economics" of SPOT requires high upfront investment in developing a large database of questions which can then be used over a number of years at low maintenance cost. The cost-effectiveness of the tool clearly derives from its multiple purpose (learning tool, self-assessment, fairer and cheaper online assessment etc.) and its applicability over a number of years.

For example, in the case of assessment, the benefits of SPOT can be illustrated by comparing the conventional quiz administration in ENGG1801, with the SPOT model. The conventional works as follows:

a. 3 quizzes are administered per semester, where students write answers on a piece of paper.
b. Because of the large number of students (typically 500), the quizzes are administered during the consecutive computer lab sessions in a particular week, with 10 different versions

of each quiz written for the 10 different sessions. This model fits well with the routine weekly lab sessions of ENGG1801 and is found easier than administering one large quiz for all students at the same time. Nevertheless, students have sometimes complained about lack of fairness, with quizzes in some sessions perceived to be easier than others.

c. Students sitting in any one session answer the same questions and invigilation is, therefore, important.
d. Tutors subsequently mark the quizzes one by one.

With SPOT, the ten-session quiz is retained. However, all quizzes are conducted online. The first two quizzes are multiple-choice, instantly marked by SPOT and the mark displayed to the student immediately after the quiz. Setting up the quizzes is simply a matter of selecting the DOQ category from which the questions are randomly chosen, with one quiz template applying for the whole class. This should be compared to the effort each year in the conventional system of setting up a total of 3×10 quiz versions. Because of the randomness of question selections, no two students sitting next to each are likely to be answering the same question and the potential for cheating is, therefore, much smaller. The third quiz is marked by tutors online because students are required to write a computer program to solve a problem. However, all the advantages of easy quiz set up and random selection of questions remain.

Clearly, the above observations about cost-effectiveness must be qualified by highlighting the importance of providing quality feedback which ensures that the tool operates well as a learning tool. This has an upward impact on the development and/or maintenance cost of SPOT, yet without in the least compromising the high level of cost-effectiveness with which it achieves its objectives.

CONCLUSION

Over the last decade or so, university teaching has undergone fundamental change as a result of three major developments: new teaching and learning possibilities offered by information technology; significant increases in class sizes in many faculties and a new emphasis on generic attributes of graduates. On the one hand, teachers are now required to achieve learning outcomes, both vocational and generic, for a larger number of students. On the other hand, teachers have a richer and more complex array of tools at their disposal. While lecturing remains an essential part of teaching, the teacher is no longer a "walking textbook," carrying a body of knowledge that he or she imparts on students at given times of the week. Instead, new methods of teaching which recognize the diversity of knowledge-acquisition processes are emerging, with the teacher

playing a triple role:

a) as a designer and manager of an effective structure of learning that the student can access in the course of the semester;

b) as a mentor helping students find their way through this structure, and negotiate the process of knowledge and skill acquisition in a resource-intensive environment, and

c) as an arbitrator of quality of learning, formally assessing students and critically analyzing these assessments in order to change and fine-tune the structure of learning that he or she has created.

The paper has shown that online tools can play a crucial role in all of the above three aspects of the teacher-student relationship—resource delivery, mentorship and assessment. Indeed, it appears that the introduction of a blended-learning approach, including a flexible online tool, to help first-year engineering students in learning computer programming, has been successful in improving student satisfaction with the course, re-focus the curriculum on the ability to write computer programs, and reduce the rate of student failures. In addition, the online tool has led unambiguously to an improvement in the quality of the assessment system and a reduction of its cost in a class of over 500 students. Student response, while highly positive, has emphasized the need to improve the quality of the automated feedback to student answers.

It is obviously difficult to characterize with precision the impact of a specific new learning tool or method on course quality because of a number of poorly-controlled variables from year to year. However, a convergence of quantitative and qualitative measures gives a strong indication that such online tools can play a critically positive role, provided their specific function in achieving learning outcomes are elicited as part of a learning activity map for the course.

ACKNOWLEDGEMENTS

The Dean of the Faculty of Engineering at the University of Sydney, Professor Greg Hancock, kindly provided the first funds for SPOT development. The Teaching-Improvement Fund program of the University of Sydney awarded us a grant for the extension of SPOT1 into SPOT2 and SPOT3. Sam Smith and Chi Yan Tang played a vital role in the project by writing the questions for the databases of SPOT1, SPOT2 and SPOT3. James Underwood conducted valuable quality checks on SPOT2 and SPOT3. Stephen Sheely and his superbly competent WebCT staff at the University of Sydney provided us with indispensable technical support in dealing with many rather complex issues related to the WebCT and Respondus software.

REFERENCES

[1] C.R. Graham, "Blended learning systems: definitions, current trends, and future directions," *Handbook of Blended Learning: Global Perspectives, Local Designs*. Bonk and Graham (eds), Pfeiffer Publishing, San Francisco, CA, 2006, pp. 3–21.

[2] J. Bourne, D. Harris, and F. Mayadas, "Online engineering education: learning anywhere, anytime", J*ournal of Engineering Education*, vol. 94, no. 1, 2005, pp. 131–146. http://www.asee.org/publications/jee/PAPERS/display.cfm?pdf=253.pdf

[3] N.M. Avouris, N. Tselios, and E.C. Tatakis, "Development and evaluation of a computer-based laboratory teaching tool", *Computer Applications in Engineering Education,* vol. 9, no. 1, 2001, pp. 8–19.

[4] F. Stern, T. Xing, D.B. Yarbrough, and A. Rothmayer, "Hands-On CFD educational interface for engineering for engineering courses and laboratories," *Journal of Engineering Education,* vol. 95, no. 1, 2006, pp. 63–183. http://www.asee.org/publications/jee/PAPERS/display.cfm?pdf=882.pdf

[5] J. Nguyen and C.B. Paschal, "Development of online ultrasound instructional module and comparison to traditional teaching methods," *Journal of Engineering Education*, vol. 91, no. 3, 2002, pp. 275–283. http://www.asee.org/publications/jee/PAPERS/display.cfm?pdf=693.pdf

[6] J. Rosenberg, "Assessing online student learning via Dantes test." *Virtual University Journal,* vol. 3, no. 1, 2000, pp. 1–7.

[7] J. Dutton, M. Dutton, and J. Perry, "Do online students perform as well as lecture students?" *Journal of Engineering Education*, Vol. 90, No. 1, 2001, pp. 131–142. http://www.asee.org/publications/jee/PAPERS/display.cfm?pdf=355.pdf

[8] B.K. Hodge and W.G. Steele, "A survey of computational paradigms in undergraduate mechanical engineering education," *Journal of Engineering Education*, vol. 91, no. 4, 2002, pp. 415–417. http://www.asee.org/publications/jee/PAPERS/display.cfm?pdf=740.pdf

[9] K. Christensen, D. Rundus, H. Fujinoki, and D. Davis, "A crash course for preparing students for a first course in computing: did it work?" *Journal of Engineering Education*, vol. 91, no. 4, 2002, pp. 409–413. http://www.asee.org/publications/jee/PAPERS/display.cfm?pdf=741.pdf

AUTHORS

Abbas El-Zein received his B.E. in Engineering from the American University of Beirut in Lebanon in 1986 and a Ph.D. in Computational Mechanics from the University of Southampton in the UK in 1990. He has also obtained a D.E.A. (Master's degree) in Environmental Sciences from the Ecole Nationale des Ponts et Chaussees in Paris in 1993. He has worked in the UK, France, Lebanon and Australia. He has been associate professor in the School of Civil Engineering of the University of Sydney since 2009. His research interests include geoenvironmental and computational geomechanics topics.

Tim Langrish received his B.E.(Hons) in Chemical and Process Engineering from the University of Canterbury in Christchurch, New Zealand, in 1985 and a DPhil in Engineering Science from Balliol

College at the University of Oxford in 1989. He has worked in the UK, New Zealand and Australia. He has been associate professor in the School of Chemical and Biomolecular Engineering at the University of Sydney since 2000. His research interests include Drying Technology, Mass Transfer and Computational Fluid Dynamics.

Nigel Balaam received his B.E.(Hons) in Civil Engineering from the University of Sydney, Australia in 1972 and a Ph.D. from the same university in 1978. He has been a professional officer in the School of Civil Engineering at the University of Sydney since 1978. His research has included soil stabilization and his main interest is in the development of software for geomechanics applications.